GEMS OF TCS

GRAPH COLORING ALGORITHMS

Sasha Golovnev
February 18, 2021
PREVIOUSLY...

- Exact Algorithms
- Randomized Algorithms
- Approximate Algorithms
Previously...

- Exact Algorithms
- Randomized Algorithms
- Approximate Algorithms
- Today: More examples
Map Coloring
SOUTH AMERICA
THE LAND OF OZ
Four Color Theorem

Theorem (Appel, Haken, 1976)

*Every map can be colored with 4 colors.*
Four Color Theorem

Theorem (Appel, Haken, 1976)

*Every map can be colored with 4 colors.*

- Proved using a computer.
Four Color Theorem

Theorem (Appel, Haken, 1976)

*Every map can be colored with 4 colors.*

- Proved using a computer.
- Computer checked almost 2000 graphs.
**Four Color Theorem**

<table>
<thead>
<tr>
<th>Theorem (Appel, Haken, 1976)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Every map can be colored with 4 colors.</em></td>
</tr>
</tbody>
</table>

- Proved using a computer.
- Computer checked almost 2000 graphs.
- Robertson, Sanders, Seymour, and Thomas gave a much simpler proof in 1997 (still using a computer search).

<table>
<thead>
<tr>
<th>Theorem (Weak Version)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Every map can be colored with 6 colors.</em></td>
</tr>
</tbody>
</table>
Six Color Theorem

Theorem (Weak Version)

Every map can be colored with 6 colors.

- Induction on the number of countries $n$. 
**Six Color Theorem**

**Theorem (Weak Version)**

*Every map can be colored with 6 colors.*

- **Induction** on the number of countries $n$.
- **Base case.** $n \leq 6$: can color with 6 colors.
**Six Color Theorem**

Theorem (Weak Version)

*Every map can be colored with 6 colors.*

- **Induction** on the number of countries $n$.
- **Base case.** $n \leq 6$: can color with 6 colors.
- **Induction assumption.** All maps with $k$ countries can be colored with 6 colors.
**Six Color Theorem**

<table>
<thead>
<tr>
<th>Theorem (Weak Version)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Every map can be colored with 6 colors.</em></td>
</tr>
</tbody>
</table>

- **Induction** on the number of countries $n$.
- **Base case.** $n \leq 6$: can color with 6 colors.
- **Induction assumption.** All maps with $k$ countries can be colored with 6 colors.
- **Induction step.** We’ll show that any map with $k + 1$ countries can be colored with 6 colors.
<table>
<thead>
<tr>
<th>Lemma</th>
<th>Euler's 5/19 for planar graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every map contains a country $v$ with at most 5 neighbors.</td>
<td></td>
</tr>
</tbody>
</table>
**Six Color Theorem. Proof**

**Lemma**

*Every map contains a country v with at most 5 neighbors.*
Lemma

*Every map contains a country \( v \) with at most 5 neighbors.*
Lemma

*Every map contains a country v with at most 5 neighbors.*
Graph Coloring
**Graph Coloring**

- A graph coloring is a coloring of the graph vertices s.t. no pair of adjacent vertices share the same color.
**Graph Coloring**

- A **graph coloring** is a coloring of the graph vertices s.t. no pair of adjacent vertices share the same color.

- The **chromatic number** $\chi(G)$ of a graph $G$ is the smallest number of colors needed to color the graph.
CHROMATIC NUMBER
CHROMATIC NUMBER
CHROMATIC NUMBER
CHROMATIC NUMBER
CHROMATIC NUMBER

χ(6) = 3

Chromatic number is 3
COMPLETE GRAPHS

The chromatic number of $K_n$ is $n$. 

= 
For $n > 1$, the chromatic number of $P_n$ is 2.
Cycle Graphs

For even $n$, the chromatic number of $C_n$ is 2.
For odd $n > 2$, the chromatic number of $C_n$ is 3.
Bipartite Graphs

The chromatic number of a bipartite graph (with at least 1 edge) is 2.

Fact: Given a 2-colorable graph, it's easy (linear time) to color it in 2 colors.
Applications
EXAM SCHEDULE

• Each student takes an exam in each of her courses
• All students in one course take the exam together
• One student cannot take two exams per day
• What is the minimum number of days needed for the exams?
EXAM SCHEDULE

• Each student takes an exam in each of her courses
• All students in one course take the exam together
• One student cannot take two exams per day
• What is the minimum number of days needed for the exams?
Exam Schedule

- Each student takes an exam in each of her courses.
- All students in one course take the exam together.
- One student cannot take two exams per day.
- What is the minimum number of days needed for the exams?

Alice takes **Graphs & Proofs**

Diagram:

- **Graphs**
- **Proofs**
- **Numbers**
- **Combs**
- **Project**
EXAM SCHEDULE

- Each student takes an exam in each of her courses
- All students in one course take the exam together
- One student cannot take two exams per day
- What is the minimum number of days needed for the exams?

Diagram:
- Graphs
- Proofs
- Numbers
- Combs
- Project

Notes:
- Same day
- Another day
BANDWIDTH ALLOCATION

Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?
Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?
Bandwidth allocation

Different stations are allowed to use the same frequency if they are far apart. What is an optimal assignment of frequencies to stations?

blue stations can use same frequency
OTHER APPLICATIONS

• Scheduling Problems
• Register Allocation
• Sudoku puzzles
• Taxis scheduling
• ...

Exact Algorithm for Coloring
Dynamic Programming

- Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$
DYNAMIC PROGRAMMING

- Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$
- Dynamic programming is one of the most powerful algorithmic techniques
Dynamic Programming

- Given graph $G$ on $n$ vertices, find $\chi(G)$—minimum number of colors in a valid coloring of $G$
- Dynamic programming is one of the most powerful algorithmic techniques
- Rough idea: express a solution for a problem through solutions for smaller subproblems
SUBPROBLEMS

- For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$
Try all subsets $U$
If 3 edge between 2 vertices in $U$ - ignore
Else $1 + x(G \setminus U)$
**Subproblems**

- For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$.
- Consider $S$. For any subset $U \subseteq S$, if there are no edges between vertices from $U$, we can color them all in one color, and use $\chi(S \setminus U)$ to color the rest.

$$\min_{U, \text{ s.t. no edges in } U} 1 + \chi(S \setminus U)$$
Subproblems

- For a subset of vertices $S \subseteq \{1, \ldots, n\}$ compute $\chi(S)$—the minimum number of colors needed to color vertices $S$.
- Consider $S$. For any subset $U \subseteq S$, if there are no edges between vertices from $U$, we can color them all in one color, and use $\chi(S \setminus U)$ to color the rest.

$$\chi(S) = \min_{U \text{ without edges}} \left(1 + \chi(S \setminus U)\right)$$
ORDER OF SUBPROBLEMS

- Need to process all subsets $S \subseteq \{1, \ldots, n\}$ in order that guarantees that when computing the value of $\chi(S)$, the values of $\chi(S \setminus U)$ have already been computed.
ORDER OF SUBPROBLEMS

• Need to process all subsets $S \subseteq \{1, \ldots, n\}$ in order that guarantees that when computing the value of $\chi(S)$, the values of $\chi(S \setminus U)$ have already been computed.

• For example, we can process subsets in order of increasing size.
Algorithm

\[ \chi(\emptyset) = 0 \]
Algorithm

\[ \chi(\emptyset) = 0 \]

for \( s \) from 1 to \( n \):

for all \( S \subseteq \{1, \ldots, n\} \) of size \( s \):

\[ \chi(S) =? \]
Algorithm

\( \chi(\emptyset) = 0 \)

for s from 1 to n:

for all \( S \subseteq \{1, \ldots, n\} \) of size s:

for all \( U \subseteq S \), \( U \) without edges

\( \chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\} \)

\( \chi(\{1, \ldots, n\}) \)
\textbf{ALGORITHM}

\[
\chi(\emptyset) = 0
\]

for \textit{s} from 1 to \textit{n}:

for all \( S \subseteq \{1, \ldots, n\} \) of size \textit{s}:

for all \( U \subseteq S, \ U \) without edges

\[
\chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\}
\]

return \( \chi(\{1, \ldots, n\}) \)
Running Time

\[ \chi(\emptyset) = 0 \]

FOR S FROM 1 TO n:

FOR ALL S \subseteq \{1, \ldots, n\} OF SIZE S:

FOR ALL U \subseteq S, U WITHOUT EDGES

\[ \chi(S) \leftarrow \min\{\chi(S), \chi(S \setminus U) + 1\} \]

RETURN \( \chi(\{1, \ldots, n\}) \)

\[
\sum_{s=1}^{n} \binom{n}{s} \cdot 2^s = 3^n
\]

Binomial theorem:

\[
(x + y)^n = \sum_{i=0}^{n} \binom{n}{i} \cdot x^i \cdot y^{n-i}
\]

\[
(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3
\]

\[
x = 2 \quad y = 1
\]

\[
3^n = (2 + 1)^n = \sum_{i=0}^{n} \binom{n}{i} 2^i \cdot 1^{n-i} = \sum_{i=0}^{n} \binom{n}{i} 2^i
\]
Randomized Algorithm for 3-Coloring
RANDOMIZED ALGORITHM

- Given a 3-colorable graph, find a 3-coloring

Compare: 2-coloring $\equiv$ Bipartite in poly time

Recall: 2-SAT in poly time

3-SAT NP-hard
Randomized Algorithm

• Given a 3-colorable graph, find a 3-coloring

• This problem is \textbf{NP}-hard, we’ll give an exponential-time algorithm
RANDOMIZED ALGORITHM

• Forbid one random color at each vertex
We get 2-SAT instance 2-SAT in linear time
RANDOMIZED ALGORITHM

- Forbid one random color at each vertex
- Solve 2-SAT in polynomial time
Randomized Algorithm

1.5^n times repeat:

- Forbid one random color at each vertex
- Solve 2-SAT in polynomial time

- Repeat the algorithm \((3/2)^n\) times
Approximate Algorithm for 3-Coloring
Approximate Coloring

- Given a 3-colorable graph, finding a 3-coloring is \textbf{NP-hard}
Approximate Coloring

- Given a 3-colorable graph, finding a 3-coloring is \textbf{NP-hard}

- Given a 3-colorable graph, finding an \textit{n}-coloring is \underline{trivial}
Approximate Coloring

• Given a 3-colorable graph, finding a 3-coloring is **NP-hard**

• Given a 3-colorable graph, finding an \( n \)-coloring is **trivial**

• We’ll see how to find an \( O(\sqrt{n}) \)-coloring in **polynomial time**
Graphs of Bounded Degree

Greedy Coloring

A graph $G$ where each vertex has degree $\Delta$ can be colored with $\Delta + 1$ colors.
Graphs of Bounded Degree

Greedy Coloring

A graph $G$ where each vertex has degree $\Delta$ can be colored with $\Delta + 1$ colors.
Greedy Coloring

A graph $G$ where each vertex has degree $\Delta$ can be colored with $\Delta + 1$ colors.
APPROXIMATE ALGORITHM For 3-coloring

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:
**APPARTIMATE ALGORITHM**

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

The set of its neighbors is 2-colorable $\equiv$ bipartite.

**Proof:** Assume not.

- $v$ cannot use any of these 3 colors.
- $v$ requires 4th color.

$\Rightarrow$ original graph is not 3-colorable. Contradiction! $\square$
While there is vertex \( v \in G \) of degree \( \geq \sqrt{n} \):

**Neighbors of \( v \) can be 2-colored, I can do this in linear time**
Approximate Algorithm

While there is vertex $v \in G$ of degree $\geq \sqrt{n}$:

- Color the neighbors of $v$ in 2 new colors,
- remove them from the graph
**Approximate Algorithm**

While there is vertex \( v \in G \) of degree \( \geq \sqrt{n} \):

1. Color the neighbors of \( v \) in 2 new colors, remove them from the graph.
2. All remaining vertices have degree \( < \sqrt{n} \). Color the rest of the graph using \( \sqrt{n} \) new colors.

**Degree**\( \Delta \leq \sqrt{n} - 1 \)

Recall: can color \( \Delta + 1 = \sqrt{n} \) colors.

- # used colors.
- In each iteration of while loop, using 2 new colors.
- How many iterations? \( n \) vertices in graph, removing \( \geq \sqrt{n} \) of them \( \Rightarrow \) # iterations \( \leq \sqrt{n} \)
- After loop: using \( \leq \sqrt{n} \) colors \( \leq 3\sqrt{n} \) colors.